Projets ISE : Prototypage et validation de systèmes embarqués

Organisation et rôles :

Responsable : Thomas Robert,

Encadrement : Thomas Robert, Laurent Pautet, Etienne Borde.

Liste de diffusion : ise@infres.enst.fr (peut être encore indisponible)

Le site web :www.infres.enst.fr/~trobert/ise2010/ (mise en place d'un service de dépôt sécurisé des livrables).

Experts techniques : Thomas Robert, Laurent Pautet.

Responsable des Audits, suivi de projet : Etienne Borde.

Le but des projets ISE est de vous placer dans une situation de développement de prototype relativement avancés de systèmes embarqués temps réels. Le but ici est de vous faire mettre en pratiques les concepts et la théorie vue en cours de FSET et ETER.

Les points abordés lors de cette instance seront les suivants :

1) Environnement de développement : Ada Ravenscar, contrôle de systèmes, configuration d'applications temps réels et serveurs de tâches. Application au Segway lego (introduire si possible le cas de la rotation).
2) Environnement de caractérisation de système embarqué : injection de faute guidée par les modèles, génération et intégration d'injecteurs de fautes prédictibles. Le cas des supports partitionnés avionique POK et/ou Xtratum.
3) Coordination de mouvements pour systèmes embarqués : développement d'un service de coordination entre N robot lego mindstorm pour un déplacement en flottilles leader – followers. Modèle train de véhicules avec vitesse et distance réglable. Maintient du statut de chaque train de véhicule et retour sur une console distante.

Environnement Ada Ravenscar pour Lego MindStorm

Etude de cas : Segway (4 personnes)

Le but de ce projet est avant tout de mettre en place, tester et valider la plate-forme de développement Ada Ravenscar for NXT sur un cas concret de développement. Les points durs seront la mise en place de la chaine de production mais aussi l'inventaire des drivers et bibliothèques portées pour le runtime Ada.



Ce sujet est organisé en 5 étapes clés dont deux sont jugées obligatoires :

1) **La mise en place d'un environnement de développement / déploiement fonctionnel**
   a. sous Windows pour Ada Ravenscar via les commandes de chargement
   b. sous Linux via de l'émulation ou l'utilitaire Wine
   c. Si b) est un succès construction d'un liveCD ou une image ISO pour mémoriser le configuration, l'installation des compilateurs, et les scripts de chargement installés dans un répertoire prédéfini de la distribution.
2) **L'implémentation de l'étude de cas en Ada Ravenscar du code de contrôle du Segway. (En mono tâche, déplacement ligne droite)**
3) Développer une interface et le module NXT associé pour piloter le robot à distance via une application déployée sur Nokia N770 entre autres (via

bluetooth, suggestion développez en Java). Offrir une interface graphique simple permettant de contrôler les paramètres du contrôleur.
4) Définir et mettre en œuvre une politique de changement de modes fonctionnels permettant de passer du cas mouvement ligne droite à la rotation…
5) Expérimenter le module dans des situation adverses (plan incliné, sol inégal …)

Livrables à échéance :

- Une image de type iso ou une installation déployable facilement sur un système Unix ou Windows de l'environnement de développement.
- Le code du prototype de segway en Ada ou C en fonction des obstacles et succès rencontrés.
- Un tutoriel complétant les manques de celui de Adacore sur le déploiement d'une application Ravenscar sur Lego MindStorm
- Un descriptif des expériences réalisées et analyse de leur déroulement.
- Le code de la commande à distance du robot.

Lien et références bibliographiques :

[1] http://www.nxtprograms.com/NXT2/segway/index.html
[2] http://libre.adacore.com/home/academia/mindstorms/
[3] http://en.wikipedia.org/wiki/PID_controller
[4] http://esweek09.inrialpes.fr/tutorials/EMSOFT%202009-Mindstorms-Tutorial-Friday-Draft.pdf
[5] http://www.newlc.com/en/nitdroid-demo-gnulinux-android-nokia-n770-0
[6] http://cgi.cse.unsw.edu.au/~cs4411/wiki/index.php?title=Segway_Control_Algorithms
[7] http://tablets-dev.nokia.com/

Fraction de développement commune : liaison bluetooth – station hôte

Injection de fautes dirigée par les modèles : application au cas des systèmes partitionnés

La caractérisation d'un support d'exécution a toujours posé un nombre de problèmes particulièrement complexes à traiter :

- Comment peut on évaluer l'efficacité du système d'exploitation d'un point de vu temporel
- Quelle confiance puis-je avoir dans les services que le système est censé me délivrer…

La méthode utilisée pour répondre à ces questions repose toujours essentiellement sur le test. Cependant, le test d'un système d'exploitation ne se fait pas de la même manière que celui d'un OS. En particulier, le traitement de la seconde question est réellement à part.

Première tranche du projet : Développer une architecture d'injection de faute reposant sur QEMU et les device drivers permettant d'implémenter les modèles de fautes suivants :

- Random bit flip pour un segment de mémoire, pour une adresse fixée
- Altération de la pile d'un thread
- Altération des valeurs de retour d'un thread par sélection de valeurs redoutées
- Injection de retards dans une exécution par busy loops.

Il faudra déterminer dans quelles mesures il est possible de sélectionner et configurer ces comportements sans avoir nécessairement à recharger le driver dans Qemu.

En fonction, du degré d'avancement, nous testerons d'abord cette architecture sur le cas d'un OS partitionné développé ici à Telecom ParisTech.

Seconde tranche du projet : Développer une grammaire simplifier permettant de spécifier une campagne d'injection de faute, les classes d'observations à collecter, et la charge appliquée au support d'exécution (pour sensibiliser les services corrompus par exemple). En fonction de la démarche de configuration du device drivers injectant les fautes, vous devrez générer automatiquement soit la configuration, soit les scripts – programmes qui l'appliquent à Qemu.

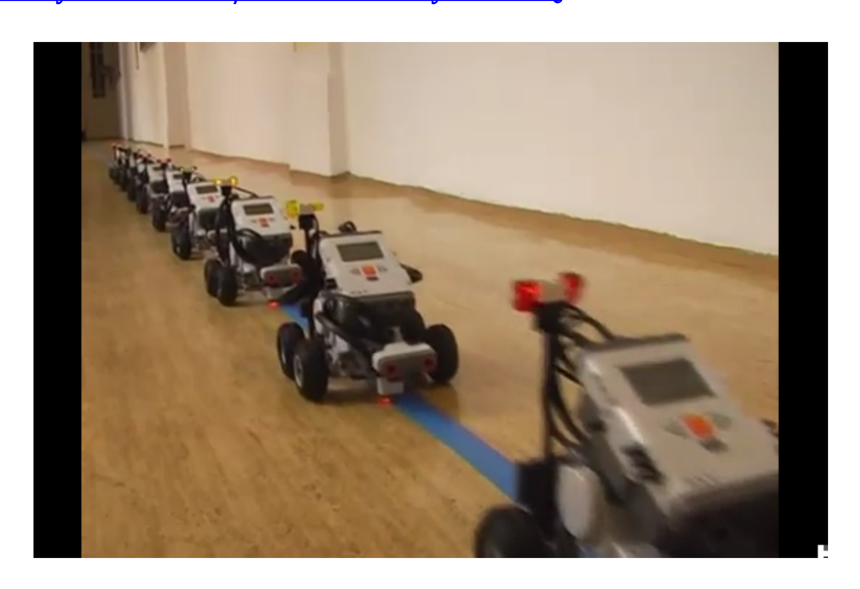
Livrables :

- La description architecturale des injecteurs de fautes en AADL pour Qemu.
- Le code métier d'injection pour une famille non triviale de fautes (3 parmi celles citées dans le sujet) pour POK et/ou Xtratum.
- Le manuel d'utilisation des injecteurs
- La grammaire du langage dédié permettant de spécifier une campagne d'injection de fautes et les observables désirés.
- Un tutoriel montrant comment utiliser l'outil de A à Z.

Références : 3 documents cf http://www.infres.enst.fr/~trobert/ISE/2010/FT/

Coordination et mouvement de groupe pour lego NXT

Le but de ce dernier projet est de mettre en place une couche réseau prédictible permettant d'implémenter un comportement de déplacement en flottille avec partage d'un état global au sein du groupe :
http://www.youtube.com/watch?v=X_siyEVTKFQ



Le problème technique est le suivant : une brique NXT communique par bluetooth avec la contrainte suivante 1 seule communication en entrée N communication en sortie avec commutation (donc inefficacité). Nous souhaitons passer outre cette limitation de ces systèmes embarqués pour implémenter un algorithme de suivi de l'état d'une flottille de robots.

Le travail à réaliser est le suivant :

1. Réaliser un prototype de robot suiveur de ligne capable de détecter la présence d'un robot juste devant lui.
2. Implémenter le contrôleur de vitesse qui tente d'assure que la valeur moyenne de l'écart entre deux véhicule est égal à L (à déterminer expérimentalement)
3. Concevoir et valider formellement le protocole permettant de créer un arbre couvrant tel que sa seul feuille soit le leader du convoi
4. Implémenter l'algorithme de construction des tables de routage. (utiliser une base si nécessaire.)
5. Implémenter sur le leader la fonction de calcul de la taille du convoi et d'export vers une base de cet état.
6. Tester et stresser le service distribué ainsi implémenté.

Livrables :

- Cahier de spécification du protocole, avec éléments de vérification (terminaison, absence de deadlock). Favoriser une première implémentation non centralisée.

- Implémentation du protocole sur brique NXT 2.0 sous Lejos ou OSEK NXT C (à définir).
- Mise en place d'une démonstration (pour une démo devant des industriels de l'automobile - Renault département R&D logiciel embarqué - en fonction du succès du projet)

Références :

1) http://www.irisa.fr/alf/index.php?option=com_content&view=article&id=57
2) http://perso.ens-lyon.fr/isabelle.guerin-lassous/Enseignement/ad-hoc-routage.pdf
3) http://lejos.sourceforge.net/forum/viewtopic.php?t=1964&sid=6b5c1c11d3e2a2ddb82e18706d5622d9